Der ISD2560

Ein Sprachchip für alle Rückmeldungen

von Wolfgang Back

Als Lallus konzeptionell entstand, war mir der Sprachchip ISD2560 noch gar nicht bekannt. Erst durch eine Recherche im Internet stieß ich auf eine Applikation. Zunächst war angedacht, eine Maschine zu bauen, die es ermöglicht, durch das Telefon gesendete DTMF-Töne zu dekodieren.

Das war am Anfang schon schwer genug, wenn man keine Hilfe von außen hat. Jedenfalls entdeckte ich irgendwann den MITEL-Chip 8870, der genau für das Detektieren von DTMF-Tönen entwickelt wurde. Als der Chip endlich richtig arbeitete, kam bald die Frage auf: „Ich sitze neben der Schaltung, versende DTMF-Töne, sehe, was passiert. Wie sieht das aber aus, wenn irgendwann einmal die Empfangsstation irgendwo angerufen wird. Woher weiß man dann, ob das alles funktioniert?“.

Es wäre nämlich fatal, wenn man einen Wert zum Rollladenheben schickt und die Sprenkleranlage geht an. Es mußte also eine akustische Rückmeldung geschaffen werden. Zunächst kam die „riesige“ Idee, einen billigen Soundchip einzusetzen, der vier verschiedene Tiergeräusche produzieren kann. Signal1 erzeugt ein Hundegebell, Signal2 produziert ein Katzengejaule, Signal3 entspricht einem Pferdewiehern und Signal4 läßt einen Hahnenschrei erahnen. Für kurze Zeit war diese Rückmeldungsmethode sogar aktiv und machte viel Freude. Perfekt war es jedoch keinesfalls, denn man mußte jedem Tiergeräusch eine Aktion zuordnen. Die jeweilige Bedeutung konnte man leicht vergessen.

Dem Treiben hat dann ein kurzer Aussetzer in der Elektronik ein Ende bereitet. Am Anfang arbeitete ich noch nicht mit einem Lineinterface, das perfekt die Schnittstelle Telefon / Elektronik bedient. Ich hatte lediglich einen Transistor mit 600 Ohm –Widerstand zum Schalten der Telefonleitung benutzt. Daß dies eine Woche gut gegangen ist, ist sowieso verwunderlich. Schließlich jedoch hat es mich erwischt. Die schöne Elektronik hat ihren Geist aufgegeben, die Tierstimmen waren zerschossen und die CControl hatte ein paar funktionierende Ports weniger.

Nachdem mittlerweile das Lineinterface installliert war, war das Problem mit der Rückmeldung der gesendeten Aktionen immer noch nicht gelöst. Schließlich kam ich auf den Gedanken, den Beep-Ausgang der CControl auf die Leitung zu legen. Schickte man per DTMF eine 4, so kam als Rückmeldung tatsächlich hörbar ein beep beep beep beep. Immer schön mitzählen hiess es da. Bei einem Doppelkreuz (12) war das schon recht störend. Waren es 11 oder 12?

Diese Methode stellte sich auch nicht als praktikabel heraus. Schön wäre es doch, wenn ich eine 4 sende, und die Gegenstelle bestätigt mir dies im Klartext, sie spricht eiune 4. Noch schöner wäre es, wenn evtl. Aktionen, wie Rollladen oben oder Rollladen unten ebenso mitgeteilt werden könnten.

Jemand kam auf die Idee, dazu einen gesteuerten Cassettenrekorder zu benutzen. Abgesehen von der Mühe, die ein solches Projekt bereitet hätte, wäre das letzte Segment von ca. 50 irgendwann einmal vorgespult gewesen. Das war alles nichts.

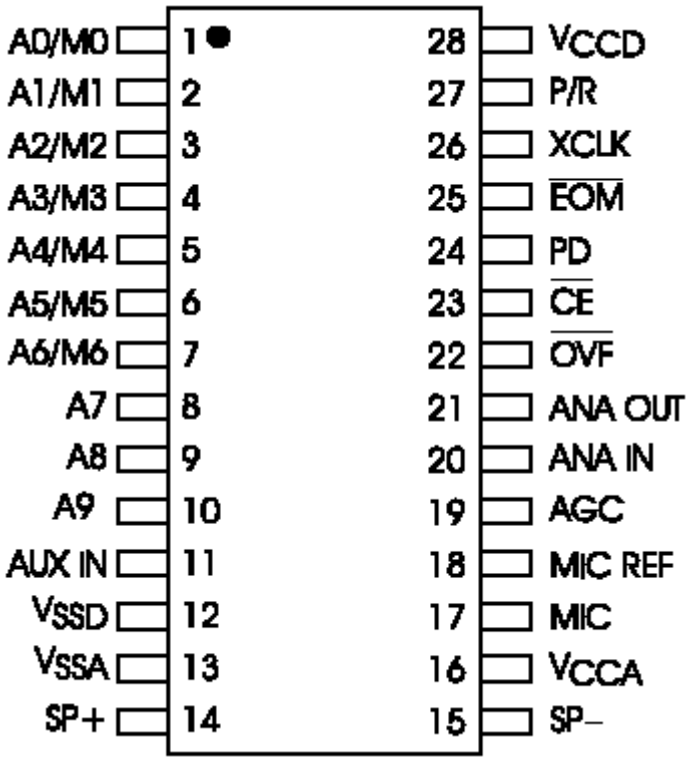
Ich verfiel dann auf einen Sprachausgabechip, den es bei Conrad Electronic gab. Zunächst einmal hatte ich Mühe, den SMD-Chip einwandfrei auf die Platine zu „kleben“. Danach stellte sich heraus, daß dieses Ding völlig unbrauchbar war. Sollte dieser Bausatz immer noch im Programm sein, so sollten Sie die Finger davon lassen. Vor allem: wenn man mühsam ein Wort eingesprochen hatte und die Betriebsspannung wurde weggenommen, so war alles weg. Also neu trainieren.

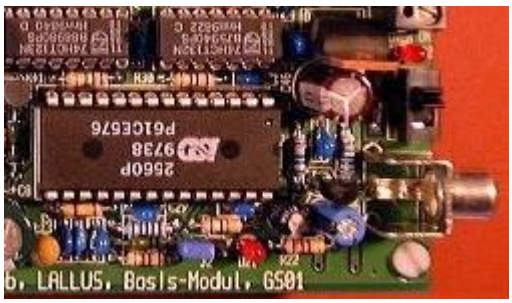
Dann -endlich- stieß ich auf den richtigen Chip von ISD. Diese raffinierte Technologie läßt es zu, den gespeicherten Inhalt ohne Betriebsspannung ca. 100 Jahre zu erhalten. (Ehrlich gesagt: ich habe es noch nicht überprüft, ob es wirklich 100 Jahre sind und nicht etwa nur 99).

Der zweite Vorteil liegt darin, daß man nicht die genaue Adresse kennen muß, an der ein gespeichertes Sprachsegment beginnt. Die Organisation übernimmt der Chip selbständig.

Ein weiterer Vorteil ist die hohe Speicherqualität des Audiosignals. Für den Betrieb über Telefon ist dieses völlig ausreichend. Die Länge der zu speichernden Segmente ist vom gewählten Chip abhängig. Es gibt hier verschiedene Ausführungen von 16 Sekunden bis 60 Sekunden in hoher Speicherqualität. Verzichtet man darauf, so kann man auch längere Passagen speichern.

Der Nachteil: die Chips sind nicht gerade billig. Der vorgesehene ISD2560 mit einer Minute Speicherkapazität kostet immerhin fast 50 Mark.





Wenn man als ungeübter Bastler das Schaltbild zum ersten Male sieht, so mag man über die Komplexität erschreckt sein. Doch später stellt sich heraus, daß es kaum etwas Einfacheres zur Ansteuerung gibt.

Der große Vorteil des ISD-Chips liegt darin, daß man einen Modus wählen kann, der mit dem Schnellvorlauf eines Cassettenrecorders zu vergleichen ist, um gespeicherte Segmente anzusprechen. Natürlich geht es hier viel schneller.

Man muß sich den Chip als einen Schrank mit gefüllten Schubladen vorstellen. Die Schubladen können sogar verschieden groß sein; sie sind lediglich in einer Reihe geordnet: entweder von oben nach unten oder von links nach rechts.

Suche ich jetzt den Inhalt der 20. Schublade, so zähle ich einfach ganz schnell von links 1,2,3,4

usw. bis zur 19. Schublade durch, ohne sie zu öffnen. Die 20. Schublade öffne ich dann und erhalte das gewünschte Ergebnis.

Bei dem Chip ist dies so gelöst, daß nach jedem aufgenommenen Segment eine elektronische Marke gesetzt wird, eom (end of message) genannt. Dieses eom kann man später im Programm sehr gut benutzen, um festzustellen, wann eine Passage zu Ende ist. Es werden nämlich Segmente verschiedener Länge gespeichert: so ist z.B. ‚aus' kürzer als ‚eingeschaltet'.

Damit wird das wahr, was ich oben angedeutet habe: das Programm zur Steuerung des Sprachchips ISD2560 ist wirklich einfach.

```
define skip  port[1]
define reset port[2]
define play  port[3]
define eom   port[4]
define i  byte
define j  byte

deact eom
reset = off : play  = on : skip  = on

for j=1 to 20

pulse reset
pause 2
skip=on
for i=1 to j
pulse play
next i

skip=off
pulse play
wait not eom
pause 3

next j
```

Dieses Programm spricht die ersten 20 gespeicherten Einträge im Sprachchip. 4 Digitalports müssen wir zur Bedienung opfern.

skip (port1) ist der Modus, der oben angesprochen wurde. Ist `skip=on`, dann läuft man an den Segmenten im Schnelllauf vorbei. Ist `skip=off`, dann ist dieser Modus ausgeschaltet.

reset (port2) setzt den Chip auf den Anfang zurück. Er steht nach diesem Befehl an Stelle 0. Damit kann man immer definiert einen Anfang setzen.

play (port3) veranlasst den Chip ein gespeichertes Segment an den Ausgang zu geben. Es genügt hier ein Anstoß, um dies zu erreichen. Damit läßt sich das Segment mit einem pulse-Befehl ausgeben.

eom (port4) wurde oben bereits erwähnt. Nach einem gesprochenen Segment wird kurzzeitig für ca. 0,8 sec ein Signal gesetzt. Dieses Signal kann man mit dem Mikrocontroller abwarten, bevor man neue Aktivitäten unternimmt.

Damit ist dieser Chip schon fast erklärt. Etwas problematisch wird noch das Einspielen der selbstgesprochenen Segmente sein. Am Anfang habe ich dies mit Tastern realisiert. Alles musste mucksmäuschenstill sein. Dann ging es los: Taster drücken, „eins" sprechen, loslassen, Taste drücken, „zwei" sprechen ..... Alle Segmente gelangten so in den Chip hinein. Manchmal war es Glückssache, daß man dem einen oder anderen Ton nichts abgeschnitten hatte. Manchmal war der

gespeicherte Ton auch zu lange. Teure Speicherverschwendung.

Aus dieser Not wurde dann das Programm **LISDREC.EXE** geboren. Es ist ein in Visual Basic 3 (16 Bit) geschriebenes Programm, das die mühsame Arbeit des Handeinspielens abnimmt.

Recht bequem kann man die gewünschten Sprachsegmente über die Soundkarte des Computers in der gewünschten Qualität aufzeichnen. Danach beginnt der Schnitt über ein Stückchen Software.

Meistens genügt schon der beigefügte Windows – Soundrecorder. Der Anfang und das Ende wird exakt geschnitten und in ein Verzeichnis gespeichert. Nach und nach entstehen so die gewünschten Segmente. Über einen komfortablen Editor in meinem Programm kann man dann eine Playliste anlegen, in der man die einzelnen Titel in ihrer Reihenfolge ordnet.

Spielt man dann noch das Programm **lisd2560.bas** in die CControl, so kann man automatisch alle Titel in den Chip einspielen.

Eine komplette Liste aller WAV-Dateien, die bei meiner Lösung eingespielt sind, ist auf der CD vorhanden, so daß man sofort einmal probieren kann. Es sind 77 einzelne WAV-Dateien. Ich kann Ihnen versichern, daß es einige Mühe macht und viel Zeit kostet, alles zu produzieren, zu schneiden und zurückzuspielen.

Durch dieses automatische Einspielen der fertigen WAV-Dateien kann man viel Speicher im ISD-Chip sparen. Als ich dies noch von Hand machte, hatte ich zwei Soundchips vorgesehen. Es war einfach zu ungenau mit den Tastern zu arbeiten. Jetzt nehmen die 77 Sprachsegmente noch keine 40 Sekunden ein. Weitere 20 Befehle könnten eingegeben werden.

Wer die Sprachausgabe mit Assembler realisieren möchte, der kann das folgende Programm einspielen:

```
*****************************************************
* Ansteuerung des Soundchips ISD2560 mit Assembler
* Programmname ISDASM.ASM
* Compilat     ISDASM.S19
*
* BASIC        define teil byte
*              #main
*              print "Segment eingeben: ";
*              input teil
*              sys &H0101 teil
*              goto main
*              syscode "isdasm.s19"
* (c) Wolfgang Back
*****************************************************

bport   equ $01          ; Digitalport 1
bpdir   equ $05          ; Richtungsbyte

skip    equ  0           ; Port 1
reset   equ  1           ; Port 2
play    equ  2           ; Port 3
eom     equ  3           ; Port 4

teil    equ  $92         ; Wert aus Basic
merker  equ  $91         ; Hilfvariable
*****************************************************
        org  $101        ; Programmstart

init    lda  #7
        sta  bpdir       ; skip,reset,play als Ausgang
```

```
        bclr eom,bpdir   ; eom als Eingang
*******************************************************
        bset reset,bport ; Reset pulsen
        bclr reset,bport

        bset skip,bport  ; Skipmodus einschalten
        jsr  lang        ; lange Pause

        lda  teil        ; Segment aus Basic
        sta  merker

loop    bclr play,bport  ; Play pulsen
        bset play,bport  ; faehrt zur Stelle teil
        jsr  kurz        ; kurze Pause

        lda  merker      ; Hilfsvariable
        deca             ; runterzaehlen
        sta  merker      ; Hilfsvariable
        bne  loop        ; auf Null pruefen

        bclr skip,bport  ; Skipmodus ausschalten
        jsr  lang        ; lange Pause

        bclr play,bport  ; play pulsen
        bset play,bport


wait    brset eom,bport,wait ; auf eom low warten
        jsr  lang            ; lange Pause
        rts                  ; zurueck nach Basic
        end

*******************************************************

kurz    lda #2               ; accu mit 2 laden
        bra loop1            ; verzweige nach loop1
lang    lda #30              ; accu mit 30 laden

loop1   ldx #255             ; x-Register laden
loop2   decx                 ; runterzaehlen
        bne loop2            ; verzweige <> 0
        deca                 ; runterzaehlen
        bne loop1            ; verzweige <> 0
        rts                  ; zurueck zum Aufruf
```